Proposta de Trabalho de Conclusão de Curso

Estudo de caso sobre o desenvolvimento de código limpo

Instituto de Matemática e Estatística
Universidade de São Paulo

Aluno: Ruan de Menezes Costa
Orientador: Prof. Dr. Marco Dimas Gubitoso

São Paulo, Abril de 2016

1 Motivação

Durante a graduação nos acostumamos a não precisar desenvolver *software* bem escrito. Os exercícios-programa (EPs) são na maioria das vezes descartáveis, dado que com o término da disciplina, ou até mesmo antes disto, nunca mais precisaremos sequer olhar para o código daquele EP. Como consequência, optamos por não mexer mais no programa a partir do momento em que ele faz o que é pedido. Sabemos que o código está ruim, mas simplesmente não compensa gastar o tempo necessário para melhorá-lo. Por outro lado, fora da universidade, *software* de qualidade é uma necessidade. Funcionários vêm e vão, mas o código permanece, precisando ser mantido e incrementado. O custo total de um programa pode ser dividido em custo inicial (desenvolvimento) e custo de manutenção [1]:

$$custo_{total} = custo_{inicial} + custo_{\text{manter}}$$

Hoje sabemos que o custo de manutenção é o fator dominante nesta equação. A manutenção é cara pois entender código já existente é uma tarefa exaustiva e passível de erros. Fazer mudanças é fácil, desde que se saiba o que mudar e que o código não seja um empecilho. Após as mudanças, ainda é necessário testar e realizar o *deploy*. Assim, chegamos ao modelo:

$$custo_{\text{manter}} = custo_{entender} + custo_{mudar} + custo_{testar} + custo_{deploy}$$

Escrevendo código limpo podemos minimizar os custos de entendimento, mudança e realização de testes, e assim minimizar o custo total.

2 Objetivos

Este trabalho tem como objetivo o desenvolvimento de um projeto real, utilizando as técnicas descritas nos livros *Clean Code* [4], *Implementation Patterns* [1], *Code Complete* [5], entre outros, para no final termos um *software* que possa ser considerado de boa qualidade.

Em geral, a primeira solução para um problema não é a mais elegante. Neste trabalho, pretende-se mostrar a evolução do código, apontando problemas de design encontrados e suas respectivas soluções. Iremos também definir conceitos como "mau cheiro"e princípios de desenvolvimento amplamente reconhecidos pela comunidade.

3 Métodos

O projeto a ser desenvolvido constitui-se na construção de um serviço web para facilitar a captura automática de dados do governo, de modo a tornar mais prático o desenvolvimento de aplicações que usufruem destes dados. A princípio serão disponibilizados apenas dados da prefeitura de São Paulo, mas isto pode ser mudado. A prefeitura atualmente já disponibiliza diversos dados, por meio da API Olho Vivo [7], que expõe dados em tempo real sobre os ônibus da capital, o conjunto de arquivos em formato GTFS [6] (General Transit Feed Specification), que expõe dados estáticos sobre o transporte público, e a plataforma

GeoSampa [3], que oferece dados diversos. No caso da plataforma GeoSampa, apesar de os dados serem de fácil acesso para humanos, o mesmo não é verdade para máquinas. O portal não facilita o download de arquivos por scripts e os dados estão muitas vezes em planilhas ou shapefiles, o que obrigaria o programa a lidar com estes tipos de arquivos.

Para o transporte público, já foi feita uma biblioteca que integra os dados no formato GTFS com os dados da API Olho Vivo, oferecendo assim o máximo de informação para o usuário.

O serviço web será desenvolvido seguindo o padrão REST [2] (Representational State Transfer). Assim, o usuário poderá ter acesso aos dados por meio de simples requisições HTTP cujas respostas serão em formato JSON. A linguagem utilizada será Java.

4 Cronograma de Atividades

Segue abaixo as ativadades a serem desenvolvidas e o cronograma aproximado.

1. Integração da API Olho Vivo com os arquivos GTFS.
2. Definição dos dados a serem tratados.
3. Desenvolvimento de bibliotecas para a captura dos dados.
4. Desenvolvimento do serviço web.
5. Estudo da literatura associada a escrita de código limpo.
6. Elaboração do poster.
7. Escrita da monografia.

Tabela 1: Cronograma

	mar	abr	mai	jun	jul	ago	set	out	nov
1	x								
2		x							
3		x	x						
4			x	x					
5		x	x	x	x	x			
6							x	x	x
7					x	x	x	x	x

Referências

[1] Kent Beck. *Implementation patterns*. Pearson Education, 2007.

[2] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[3] LabProdam. Geosampa. `http://geosampa.prefeitura.sp.gov.br/PaginasPublicas/_SBC.aspx`.

[4] Robert C Martin. *Clean code: a handbook of agile software craftsmanship*. Pearson Education, 2009.

[5] Steve McConnell. *Code complete*. Pearson Education, 2004.

[6] SPTrans. Gtfs sptrans. `www.sptrans.com.br/desenvolvedores/Default.aspx`.

[7] SPTrans. Olho vivo api. `http://www.sptrans.com.br/desenvolvedores/APIOlhoVivo/Documentacao.aspx`.